

Hew (Ver 4.04) +KPIT (v0703) を使用した場合の

新ワークスペースおよびプロジェクトを登録する方法

(H8/3048F-ONE E10T版)

KPIT版の統合環境「Hew Ver 4.04」で H-debugger 用に新ワークスペース/プロジェクトを登録する手順方法を説明します。

説明を明確にするために、名前等を仮に決めて例に沿って説明を進めます。

ワークスペース名	KPIT3048		
プロジェクト名	Project		
登録モジュール名	H8_3048FONE.c	C	メインモジュール (アプリ用)
	KpitDebugH8_2.h	ヘッダファイル	ソフトパーツ用定義ファイル (ソフトパーツを使用しない場合は不要です。)
KPIT 添付ファイル	start.asm	ASM	スタートアップモジュール
	hwinit.c	C	ハード初期化用モジュール
	vects.c	C	リセット/割込みベクターテーブル
	inthandler.c	C	割込みハンドラー用
	sbrk.c	C	ヒープメモリー用
	iodef.h	ヘッダファイル	I/O 定義ビットフィールド記述用
	inthandler.h	ヘッダファイル	割込みハンドラー用
CPUタイプ	H8/3048F-ONE		

1. 新ワークスペースの登録方法

“HEW” 起動させます。

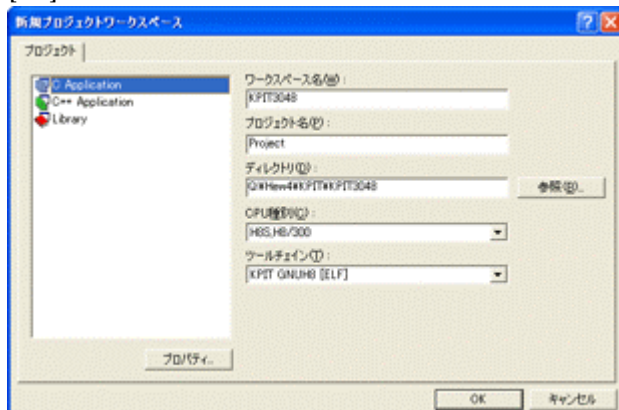
[1-1]



“新規プロジェクトワークスペース”をチェックしてのOKをクリックする。

もしくは、キャンセル後に、[ファイル]-[新規ワークスペース]をクリックします。

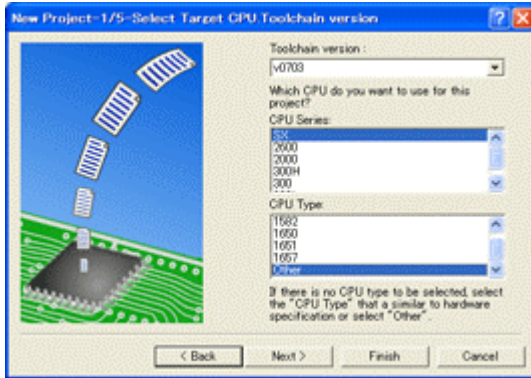
[1-2]



ワークスペース名 KPIT3048
プロジェクト名 Project
ディレクトリ C:\Hew4\KPIT\KPIT3048
CPU種別 H8S, H8/300
ツールチェイン KPIT GNUH8 [ELF]
プロジェクト Application

この項目を設定確認後OKをクリックして下さい。

[1-3]

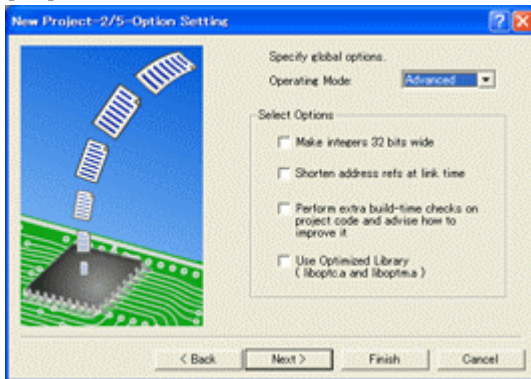


CPUスペックを選択します。

- ①300H
- ②3048F

N e x t >をクリックします。

[1-4]

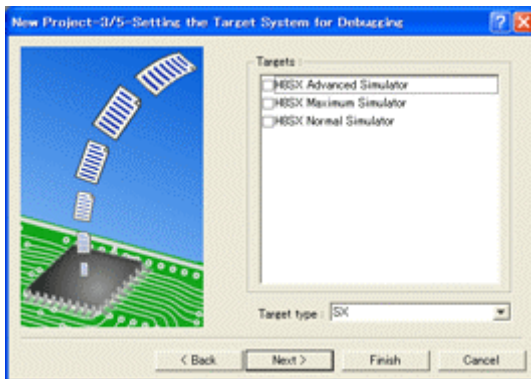


CPUオプションを選択します。

- ①「Operation Mode」<Advanced>
- ②「Use Optimized Libraries」
チェックを外します。

N e x t >をクリックします。

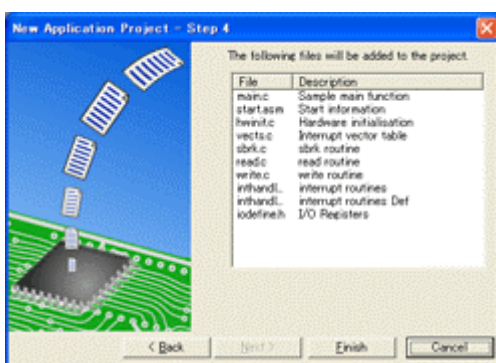
[1-5]



シミュレータの設定ですが使用しませんのでチェック無しの状態で、

N e x t >をクリックします。

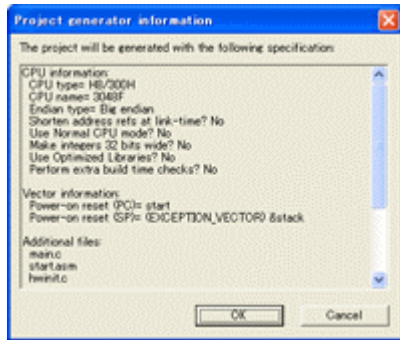
[1-6]



作成されるファイル一覧表示です。

F i n i s h >をクリックします。

[1-7]



最終確認画面です。

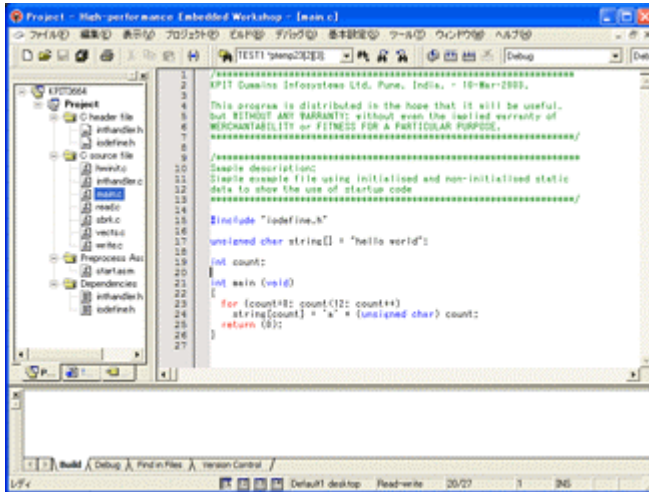
OKをクリックします。

ここまでの操作が新規プロジェクトの登録方法です。

2. プロジェクトから不要モジュール（ソースファイル）を削除します。

目的： K P I Tにより準備されたモジュールを使用しない場合に削除しておきます。

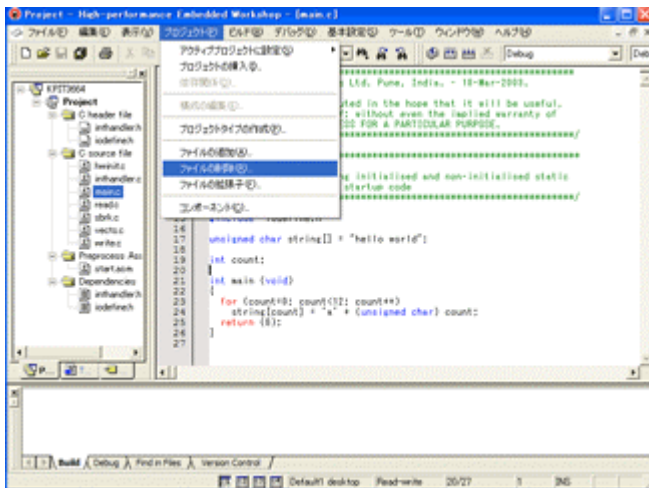
[2-1]



今回の使用例では下記1 ファイルを削除します。

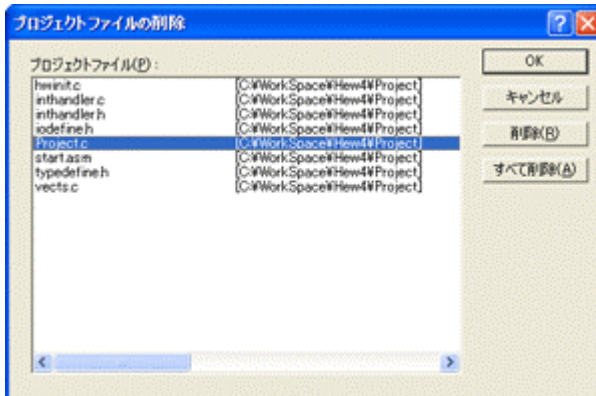
①Project.c

[2-2]



[プロジェクト] –
[ファイルの削除] をクリックします。

[2-3]

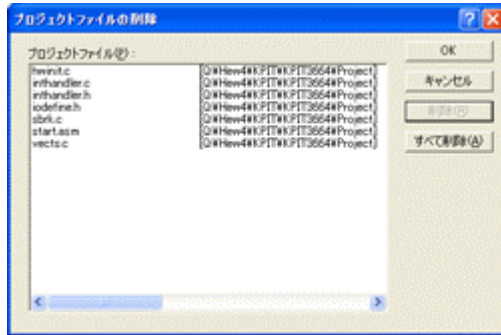


① project.c

の1ファイルを選択する。

削除 をクリックします。

[2-4]



確認画面です。

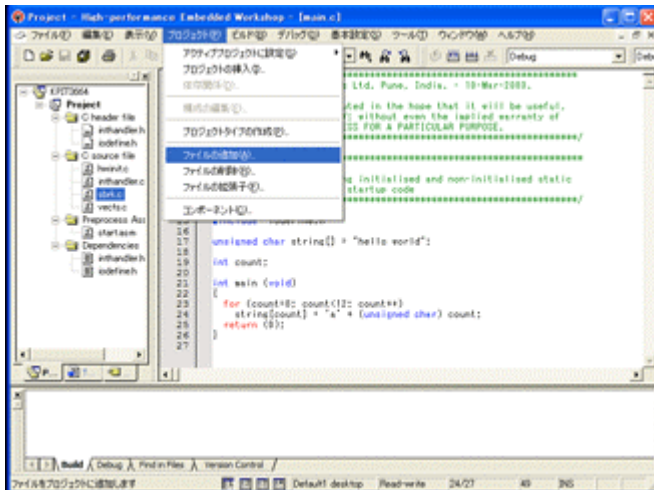
OKをクリックします。

3. プロジェクトに希望モジュール（ソースファイル）を登録します。

準備： 作成済みの2ファイルを”Q:\Hew4\KPIT\KPIT3048\Project”にコピーします。

H8_3048FONE.c HP よりダウンロードします。(GNU/gcc)
KpitDebugH8_2.h KPIT3048_v0703_1.LZH

[3-1]



[プロジェクト]-
[ファイルの追加]をクリックします。

[3-2]



登録ファイルを選択します。

① H8_3048FONE.c

追加をクリックします。

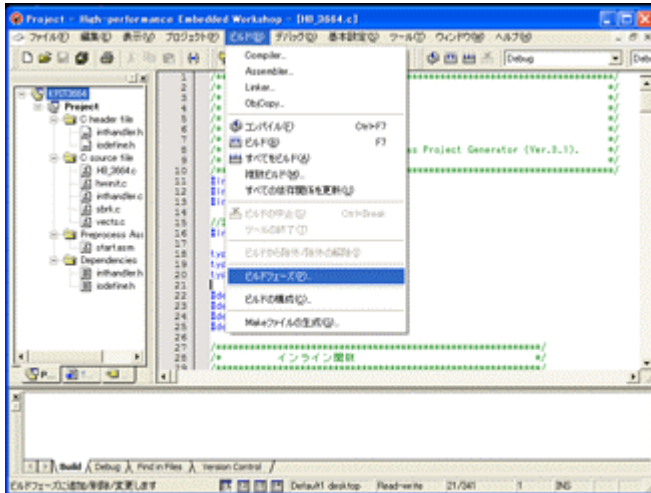
この操作によりプロジェクトにモジュールを登録します。

4. シンボルコンバータ「GCsymconv」を登録します。

目的：H-debuggerでシンボリックデバッグする為にアプソリュートファイル【Project.x】からシンボル情報抽出します。

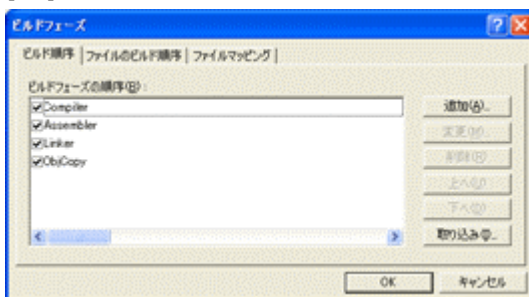
準備：KPTI用シンボルコンバータ【GCsymconv.exe】をホームページよりダウンロードし解凍後、DEFインストールDIR「C:\Program Files\Aone\DEF」下にコピーして下さい。

[4-1]



[ビルド]-
[ビルドフェーズ]をクリックします。

[4-2]



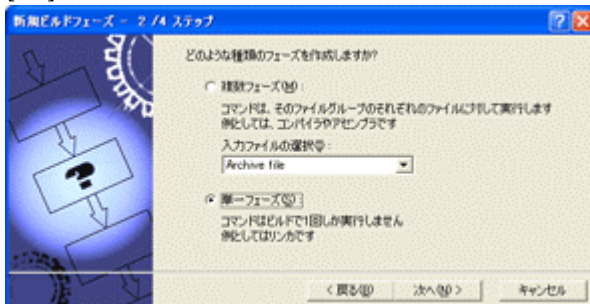
追加をクリックします。

[4-3]



次へ>をクリックします。

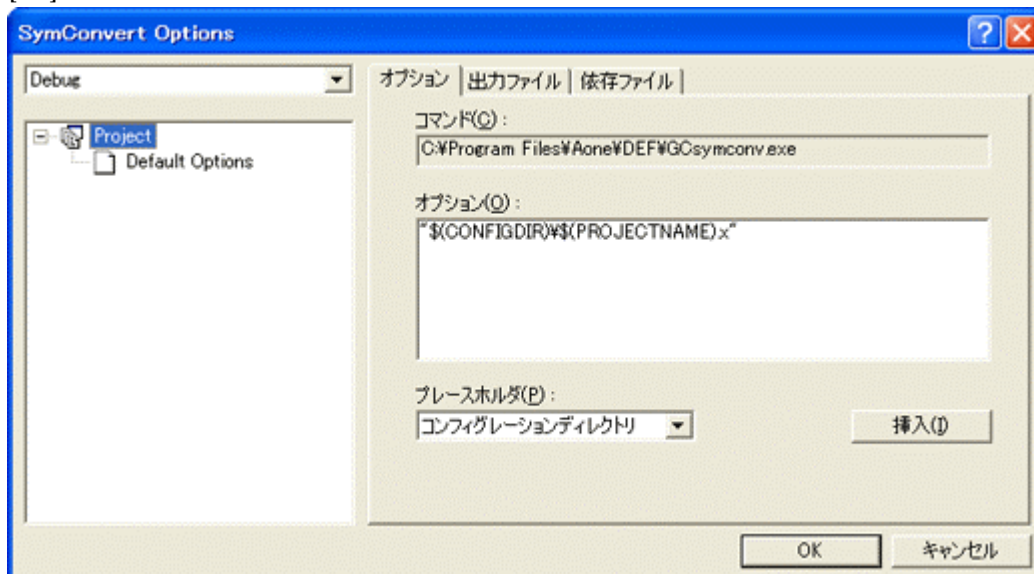
[4-4]



単一フェーズ側にチェックをします。

次へ>をクリックします。

[4-9]



①オプションに下記内容を設定する。

"\$(CONFIGDIR)\\$(PROJECTNAME).x"

(入力ファイル名)

②OKをクリックします。

注意事項

①ディレクトリ名に ' ' スペースを使用している場合は、"" ダブルクォートで囲んで下さい。

"\$(CONFIGDIR)\\$(PROJECTNAME).x"

②\$(PROJECTNAME)の先頭に「¥」記号を入力して下さい。(手入力)

③オプションSWを使用する場合は両端にスペースを入れてください。(手入力)

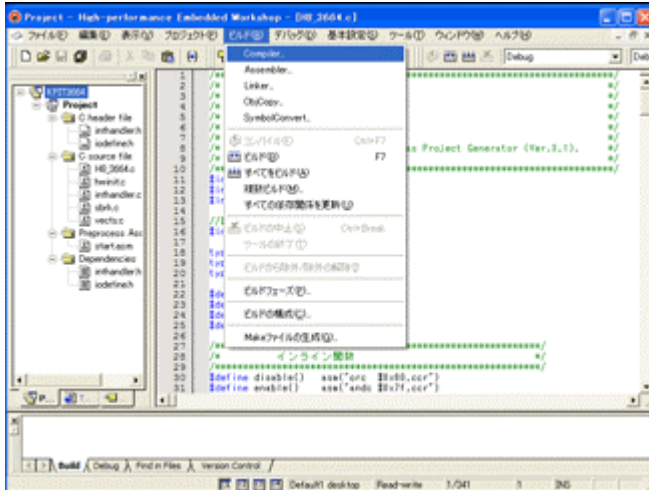
追加事項 (スイッチ説明)

- 1) [-o] (省略可) 出力ファイル名を指定
- 2) [-r] (省略可) モジュール毎のディレクトリ情報を作成しない。
- 3) [-s] (省略可) ラインシンボル情報をソート (アドレス順) しない。
- 4) [-i] (省略可) 重複モジュール情報を削除する。
- 5) [-g] (省略可) スタティック変数をグローバル化する。(Ver 1. 20xから)
- 6) [-m] (省略可) 重複モジュール情報をCソースにマージする。(Ver 1. 40Bから)
- 7) [-f] (省略可) 使用インクルードファイルをCViewに登録する。(Ver 1. 40Bから)

5. コンパイラオプションの確認と設定をします。

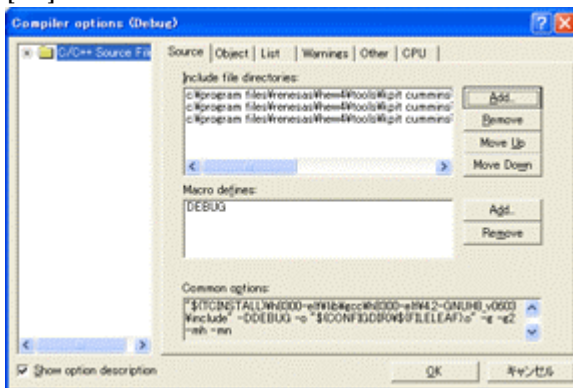
目的： H-debugger でシンボリックデバッグを可能にする為、コンパイラオプションの確認と設定をします。

[5-1]



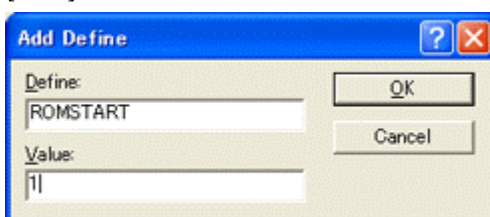
[ビルド]-
[Compiler]をクリックします。

[5-2]



[Source] タグ
① 「Macro defines」
Add をクリックします。

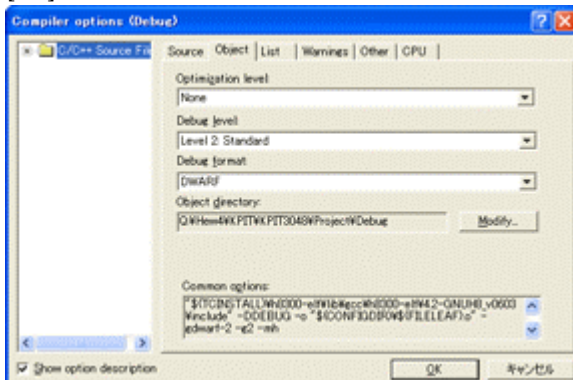
[5-2-1]



②Define: 「ROMSTART」を登録します。
③Value: 「1」を登録します。
④OK をクリックします。

初期値を ROM->RAM にコピーする為の重要な設定です。

[5-3]



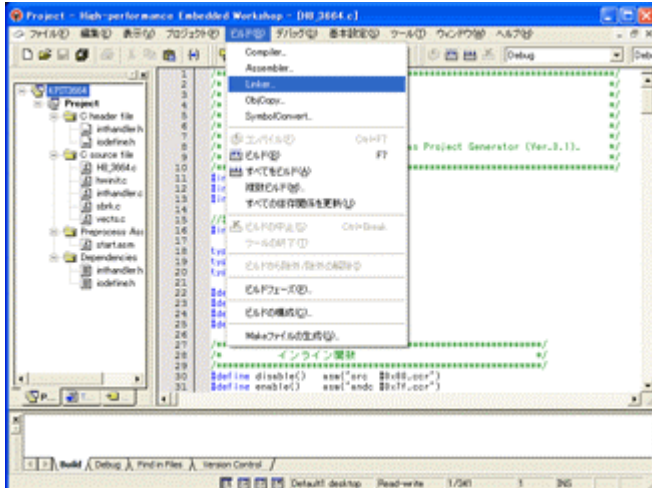
[Object] タグ
①Optimization : None(Default)
②Debug level : Level2:Standard(Default)
③Debug format: **DWARF** に指定する。
④Object directory : (Default)状態

OK をクリックします。

6. リンカーオプションの確認と設定をします。

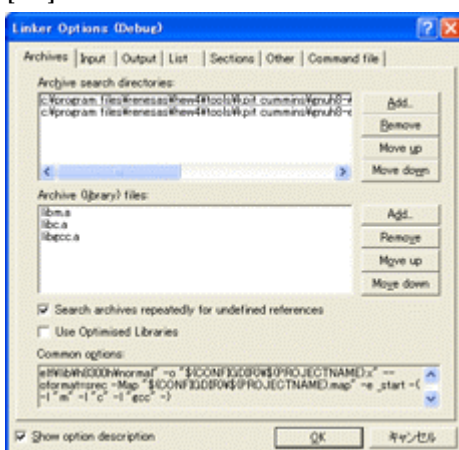
目的： H-debugger でシンボリックデバッグを可能にする為、リンカーオプションの確認と設定をします。

[6-1]



[ビルド]-
[Linker]をクリックします。

[6-2]



[Archives] タグ
デフォルト状態です。(変更の必要なし)

【HowTo】

「Archive search directories」の情報が何らかの原因により変わってしまった場合、「Use Optimised Libraries」のチェックを付けて外しますとデフォルト状態に戻ります。

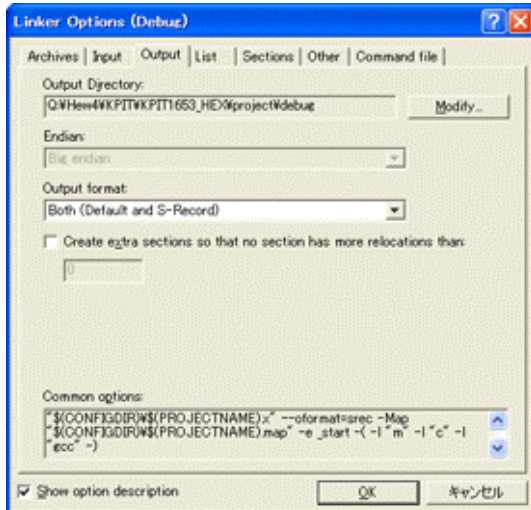
[6-3]



[Input] タグ
基本的には何も設定しなくて良いですが、各モジュールのリンク順番を指定したい場合に全モジュールをここで指定します。

- ① vects.o
- ② start.o
- ③ inthandler.o
- ④ hwinit.o
- ⑤ sbrk.o
- ⑥ H8_3048FONE.o

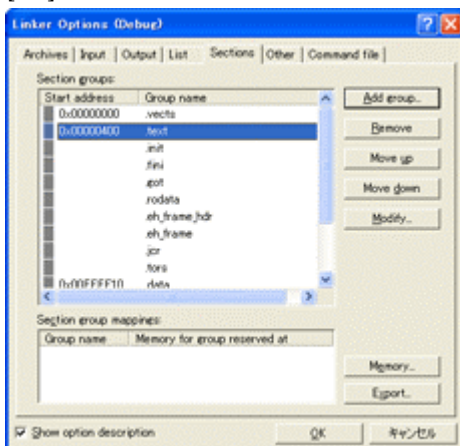
[6-4]



[Output] タグ

- ①Output Directory : (Default)
 - ②Endian : Big endian(Default)
 - ③Output format :Both(Default and S-Record)
- すべて、デフォルトです。

[6-5]

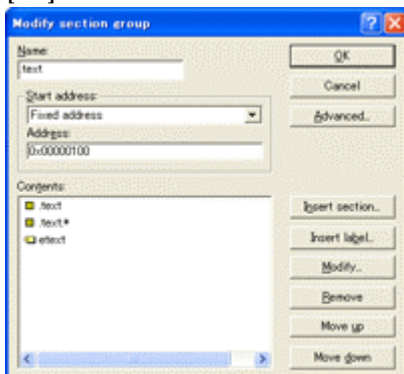


[Sections] タグ

.text セクションの開始アドレスを変更します。
デフォルトで「0x400」になっていますが、「0x100」番地から可能ですので変更します。

- ①.text セクションを選択します。
- ② **[Modify]** PB をクリックします。

[6-6]



③Address: を「0x100」に変更します。

④ **[OK]** をクリックします。

[6-7]

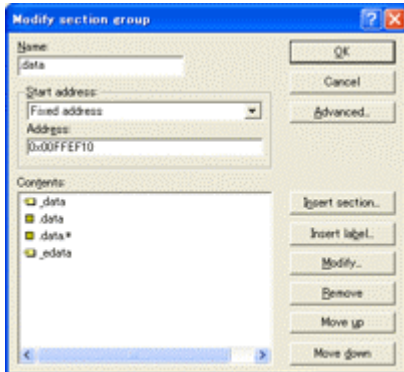


[Sections] タグ

.data セクションのアドレスを変更します。

- ①.data セクションを選択します。
- ② **[Modify]** PB をクリックします。

[6-8]

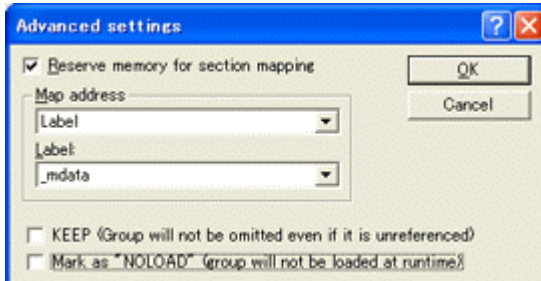


- ③Start address: **[Fixed address]** に選択します。
- ④Address: .data セクションの先頭アドレスを指定します。
[0xFFEF10]

ソースブレイク使用の指定の場合でも内臓RAMは使用しません。

- ⑤ **Advanced** をクリックします。

[6-9]



- ⑥ 「Reserve memory for section mapping」 をチェックする。
- ⑦Map address: **[Label]** にする。
- ⑧Label: **[_mdata]**
- ⑨ **OK** をクリックします。

初期値を **ROM->RAM** にコピーする為の重要な設定です。

[6-10]



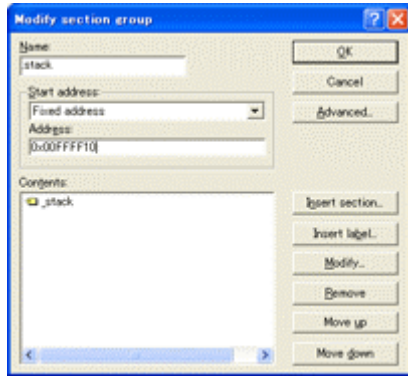
[Sections] タグ

.stack セクションのアドレスを指定します。

ここでの指定値は、スタックポインタへの初期設定値になります。

- ①.stack セクションを選択します。
- ② **[Modify]** PB をクリックします。

[6-11]



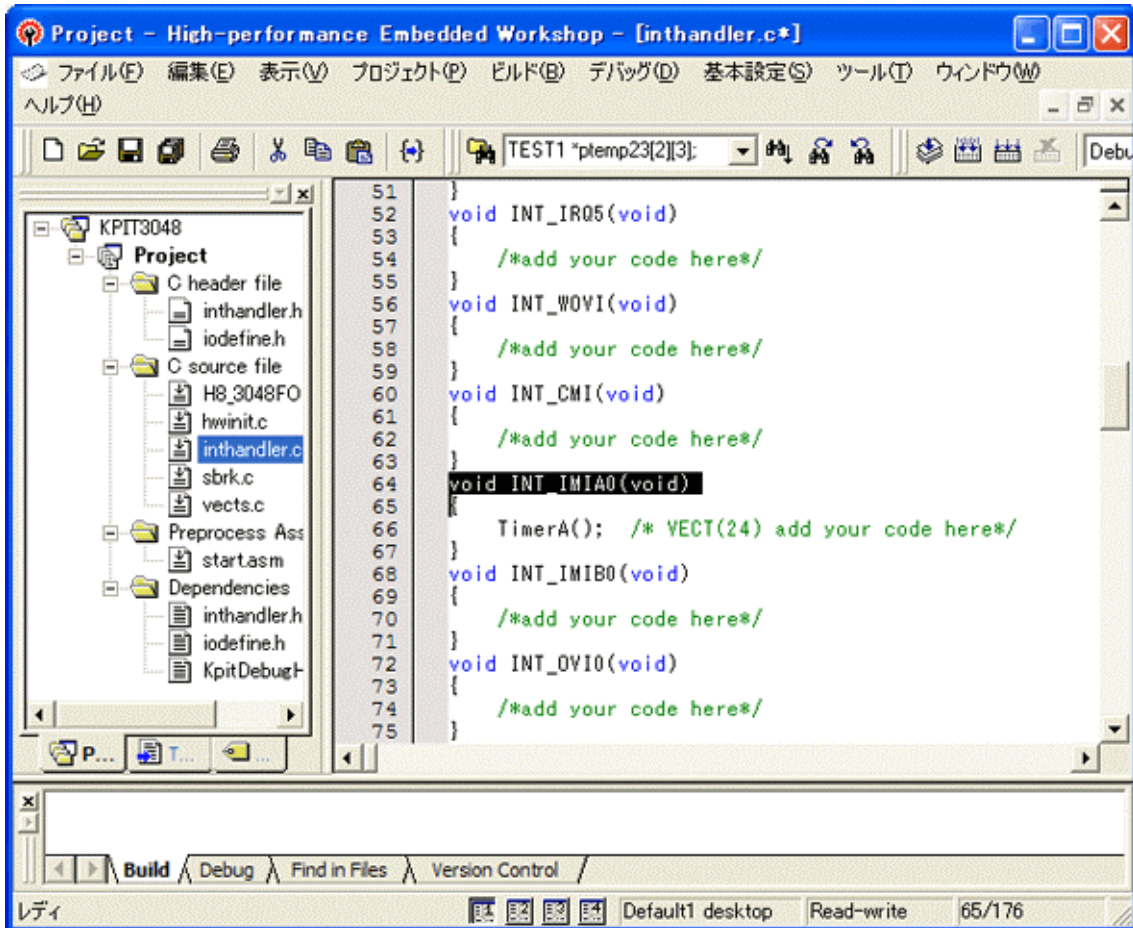
③Address: を「0x00FFFF10」に変更します。

④OKをクリックします。

7. 割り込みハンドラへ登録します。

目的： 今回説明に使用したモジュール「H8_3048FONE.c」は、TimerA（ベクター24）の割り込みを使用していますので、割り込みハンドラへの登録をします。

[7-1]



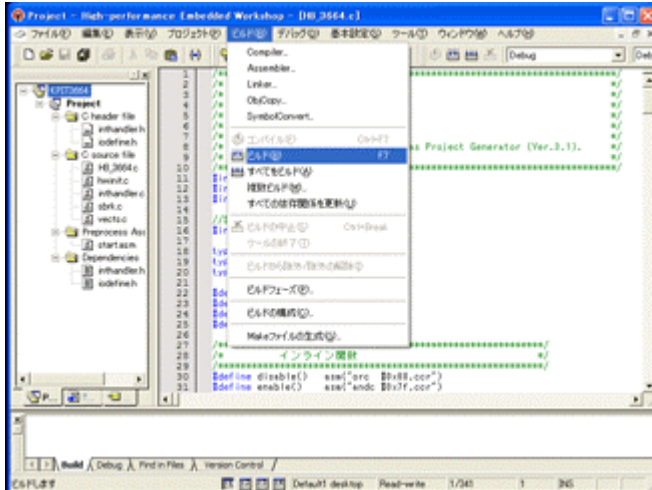
① inthandler.c を選択します。

②void INT_IMIA0(void) { TimerA(); }の関数を記述します。

8. ビルドを実行します。

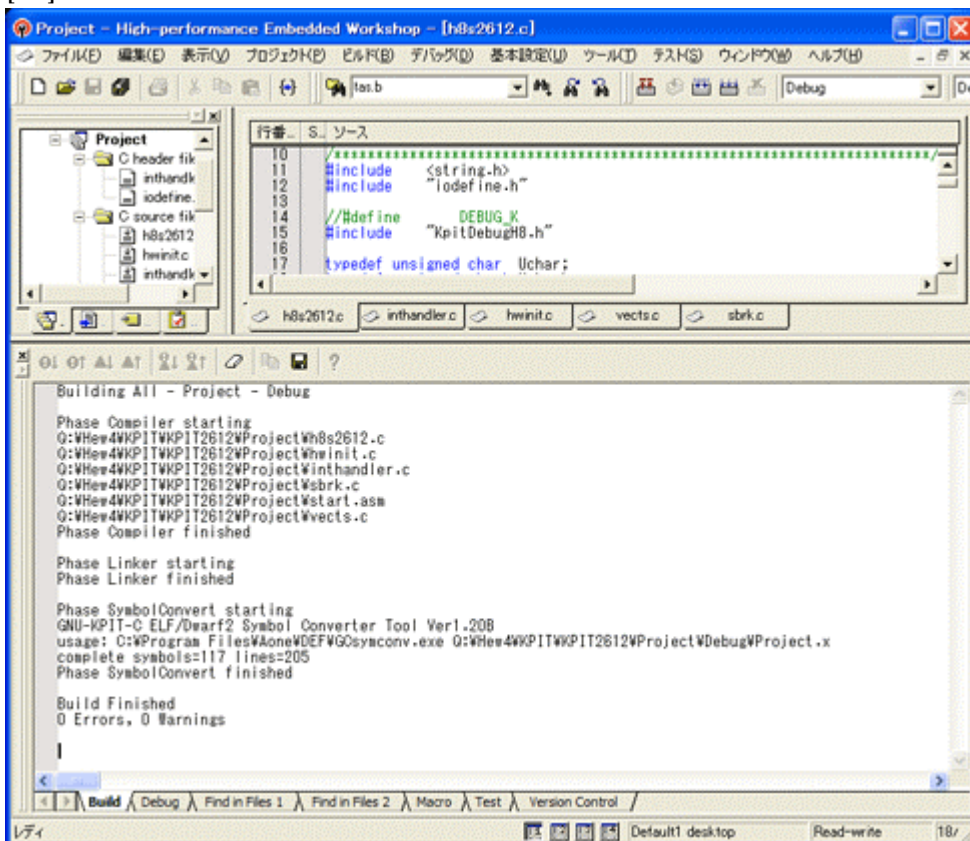
目的： コンパイル／アセンブリ／リンクロケート／GCsymconv を実行させる為、ビルドを実行します。

[8-1]



[ビルド]-
[ビルド]をクリックします。

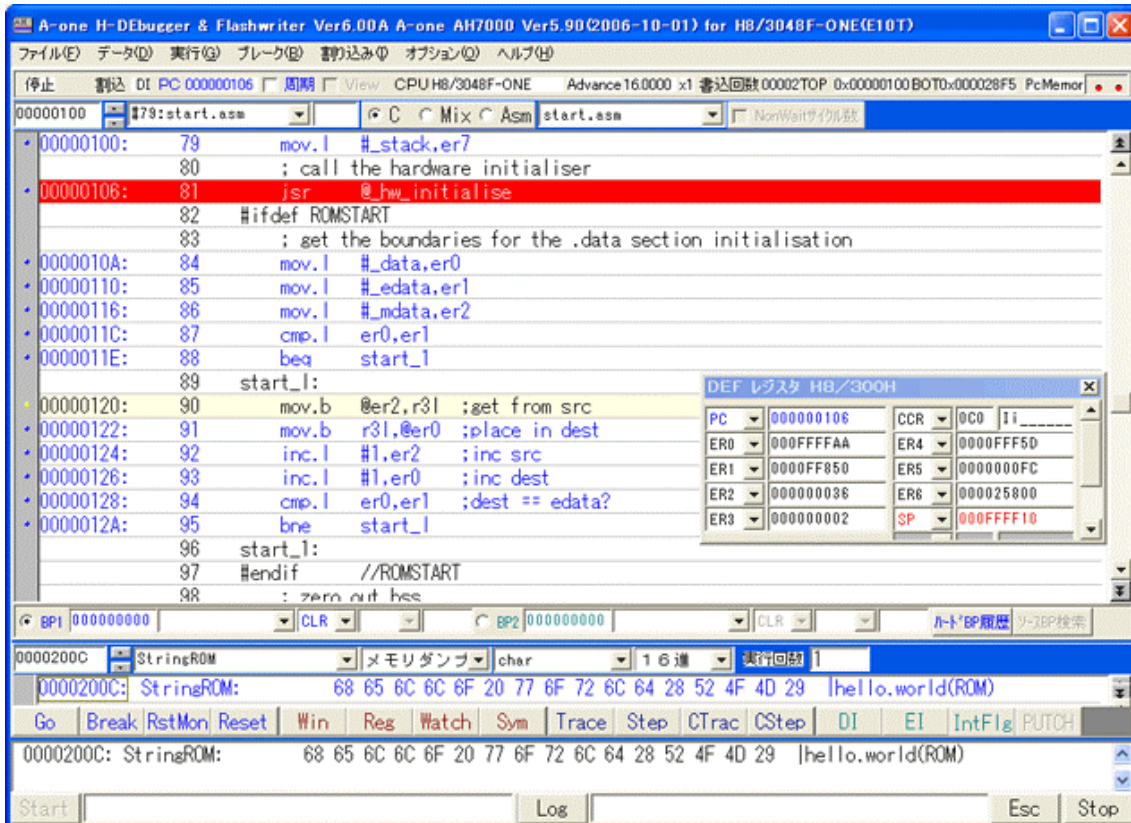
[8-2]



↑のように「0 Errors, 0 Warnings」になれば成功です。

9. DEFでの確認

[9-1]



①100H番地にスタックポインタの設定プログラムが確認できます。

これで「H-Debugger」用の設定作業が終了です。

以上